A Project Report

on

Medical Image Segmentation

*carried out as part of the **Minor Project IT3270** Submitted by*

Aaditya Sharma**, **199302117
Nikhil Bhatnagar**, **199302121

VI-IT

In partial fulfilment for the award of the degree

of

Bachelor of Technology

In

Information Technology

MANIPAL UNIVERSITY
JAIPUR

School of Computing and Information Technology

Department of Information Technology

MANIPAL UNIVERSITY JAIPUR

JAIPUR-303007

RAJASTHAN, INDIA

May 2022

CERTIFICATE

Date: 24.05.2022

This is to certify that the minor project titled **Medical Image Segmentation** is a record of the bonafide work done by **Nikhil Bhatnagar (199302121) and Aaditya Sharma (199302117)** submitted in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Information Technology of Manipal University Jaipur, during the academic year 2021-22.

Dr Pratistha Mathur

Project Guide, Department of Information Technology

Professor

Manipal University Jaipur

Dr Pankaj Vyas

HoD, Department of Information Technology

Manipal University Jaipur

Abstract

The human brain is the major controller of the humanoid system. The abnormal growth and division of cells in the brain leads to a brain tumor, and the further growth of brain tumor leads to brain cancer. CT scans, X-Ray, and MRI scans are the common imaging methods among magnetic resonance imaging (MRI) that are the most reliable and secure. MRI detects every minute objects. Our paper aims to focus on the use of different techniques for the discovery of brain tumor using brain MRI.

In this project, we make use of MRI images of brain tumor, and its correspondent mask images to train our model to generate accurate results on predicting and segmenting these tumors. We'll be using the Convolution Neural Network (CNN) 'U-Net' for achieving the same. Convolution Neural Network (CNN) segmentation techniques for reliable detection of the tumor region.

List of Figures

List of Tables

Table of Contents

1. Introduction

Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics. The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image. When applied to a stack of images, typical in medical imaging, the resulting contours after image segmentation can be used to create 3D reconstructions with the help of interpolation algorithms like marching cubes. Each of the pixels in a region is similar with respect to some characteristic or computed property, such as colour, intensity, or texture.

In digital image processing and computer vision, image segmentation is the process of partitioning a digital image into multiple image segments, also known as image regions or image objects (sets of pixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyse. Detection of brain tumor using a segmentation approach is critical in cases, where survival of a subject depends on an accurate and timely clinical diagnosis. Gliomas are the most found tumours having irregular shape and ambiguous boundaries, making them one of the hardest tumours to detect. Our system also covers Meningioma and Pituitary Tumor which are some of the most common forms of brain tumor usually found.

Problem Statement

To automate the process of segmenting brain tumor regions in MRI images by implementation of Convolutional Neural Network.

A U-Net based model that will be able to accurately locate the tumor regions in brain and perform semantic segmentation for obtaining accurate output images for the same.

For comparison purposes a CNN architecture, M-Net is also incorporated.

Objectives

- Understanding how to implement theory to practice. To choose the right Dataset for our project so that the MRI images are clear and sharp.

- Normalize the images by subtracting the mean of each feature and a division of standard deviation to obtain a suitable range of pixel values for easy use.

- To understand how to implement U-Net & M-Net architecture to build our model upon and generate accurate results by training it on the dataset. We will also be working upon specific methods to increase accuracy of our U-Net & M-Net model by applying some skip functions between the convolutional and max pooling layers.

- Finally, we aim to device a fully working neural network model to automate the process of segmentation of different tumor regions that could be present in the brain and develop it to the point where it can be used in the medical image domain for the welfare of the community.

Scope of Project

Upon completion, a fully working neural network model to automate the process of segmentation of different tumor regions will be constructed. The tumor regions that could be present in the brain will be efficiently identified by the model and will be developed to the point where it can be used in the medical image domain for the welfare of the community.

2. Background Detail

2.1. Literature Review

- **Asra Aslam, Ekram Khan, M.M. Sufyan Beg, Improved Edge Detection Algorithm for Brain Tumor Segmentation, Procedia Computer Science, Volume 58,2015, Pp 430-437, ISSN 1877-0509.**
  M. M. Sufyan et al. has presented a detection using enhanced edge technique for brain-tumor segmentation that mainly relied on Sobel feature detection. Their presented work associates the binary thresholding operation with the Sobel approach and excavates diverse extents using a secure contour process. After the completion of that process, cancer cells are extracted from the obtained picture using intensity values.

- **Mahmoud, Dalia & Mohamed, Eltaher. (2012). Brain Tumor Detection Using Artificial Neural Networks. Journal of Science and Technology. 13. 31-39.**
  Dalia Mahmoud et al. [9] presented a model using Artificial Neural Networks for tumor detection in brain images. They implemented a computerized recognition system for MR imaging the use of Artificial Neural Networks. That was observed that after the Elman community was used during the recognition system, the period time and the 9-accuracy level were high, in comparison with other ANNs systems. This neural community has a sigmoid characteristic which elevated the extent of accuracy of the tumor segmentation.

- **P.S. Mukambika, K Uma Rani, "Segmentation and Classification of MRI Brain Tumor," International Research Journal of Engineering and Technology (IRJET), Vol.4, Issue 7, 2017, pp. 683 – 688, ISSN: 2395-0056.**
  Mukambika et al. proposed methodology for the subsequent stage's classification of the tumor, whether it is present or not. Their proposed work represents the comparative study of strategies used for tumor identification from MR images, namely the Level set approach and discrete wavelength transforms (DWT) and K-method segmentation algorithms. After that phase, feature extraction is done followed SVM classification

- **Priyanka Kadam , S.N. Pawar, "Brain Tumor Segmentation and Its Features Extraction by using T2 Weighted Brain MRI", International Journal of Advanced Research in Computer and Communication Engineering ISO 3297:2007 Certified Vol. 5, Issue 7, July 2016.**
  The algorithm has been developed, which skips the areas of brain which do not suits the criteria of high intensity and high entropy, because these are the two main features of brain tumor in T2 weighted MRI. After that the image is rebuilt by using the extended maxima transformation, which helps to detect the brain tumor.

- **Anil Singh Parihar, "A Study on Brain Tumor Segmentation Using Convolution Neural Network", Department of Information Technology Delhi Technological University**
  The methods and their results have been studied for brain tumor segmentation using CNN. First Method involves brain tumor segmentation that tackles the diversity in multiple sites', multiple scanner MRI images, using the method of intensity normalization which shows that it is relevant for good segmentation using MRI images

- **Raghav Mehta, "M-NET: A CONVOLUTIONAL NEURAL NETWORK FOR DEEP BRAIN STRUCTURE SEGMENTATION", Jayanthi Sivaswamy Center for Visual Information Technology (CVIT), IIIT-Hyderabad, India**
  The Convolutional Neural Network (CNN) architecture called the M-net, for segmenting deep (human) brain structures from Magnetic Resonance Images (MRI). Consequently, the M-net utilizes only 2D convolution though it operates on 3D data, which makes M-net memory efficient.

- **Shruti Jadon, "A SURVEY OF LOSS FUNCTION FOR SEMANTIC SEGMENTATION", IEEE Member, India, shrutijadon@ieee.org**
  Various papers came up with different objective loss functions used in different cases such as biased data, sparse segmentation, etc. In this paper, she has summarized some of the well-known loss functions widely used for Image Segmentation and listed out the cases where their usage can help in fast and better convergence of a model. Furthermore, she has also introduced a new log-cosh dice loss function and compared its performance on NBFS skull-segmentation open-source dataset with widely used loss functions.

3. Methodology

3.1. System Architecture

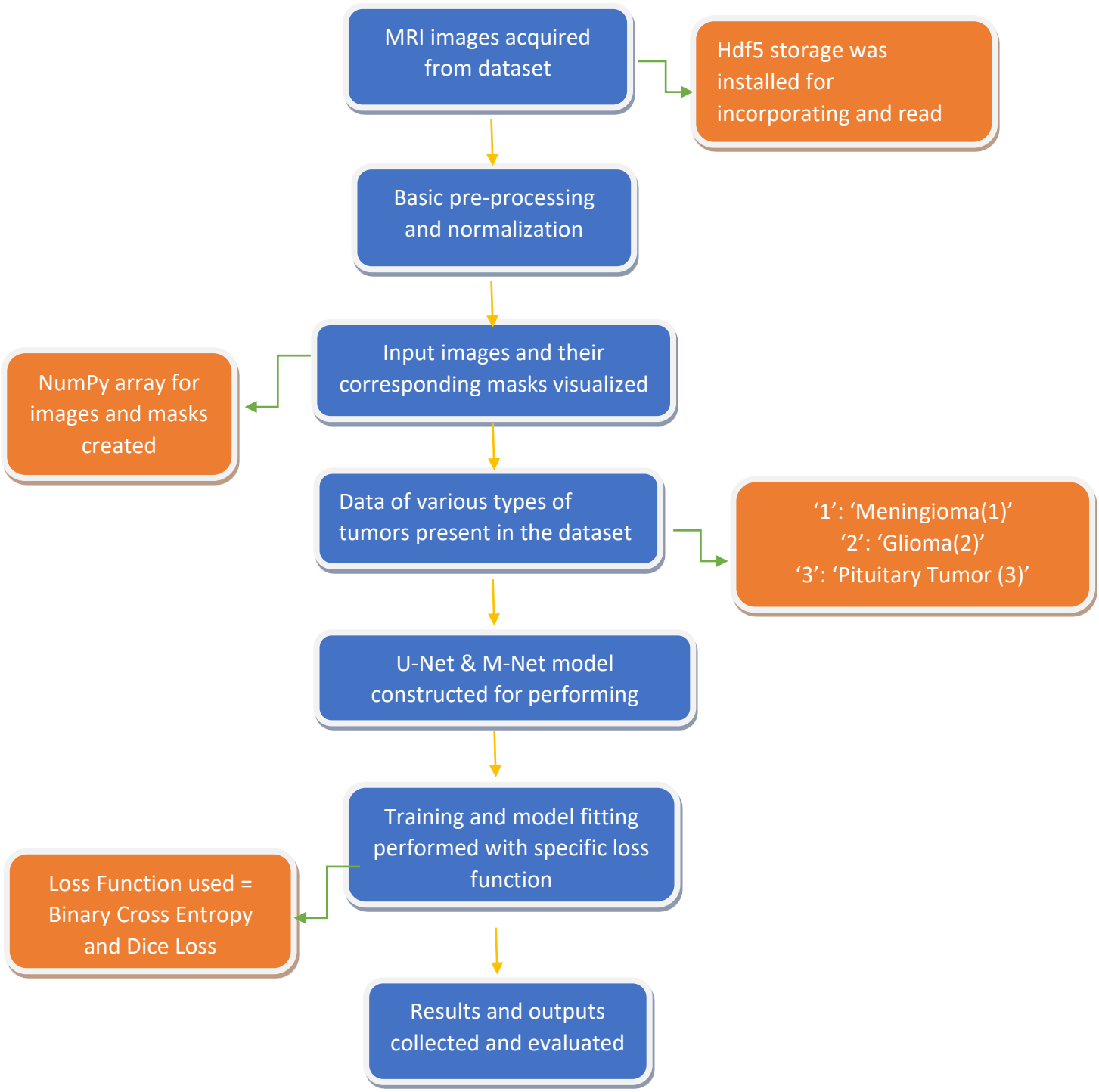


Figure 1: Flowchart for System Architecture

MRI Image acquired from dataset

This brain tumor dataset containing 3064 T1-weighted contrast-enhanced images from 233 patients with three kinds of brain tumor: meningioma (708 slices), glioma (1426 slices), and pituitary tumor (930 slices).

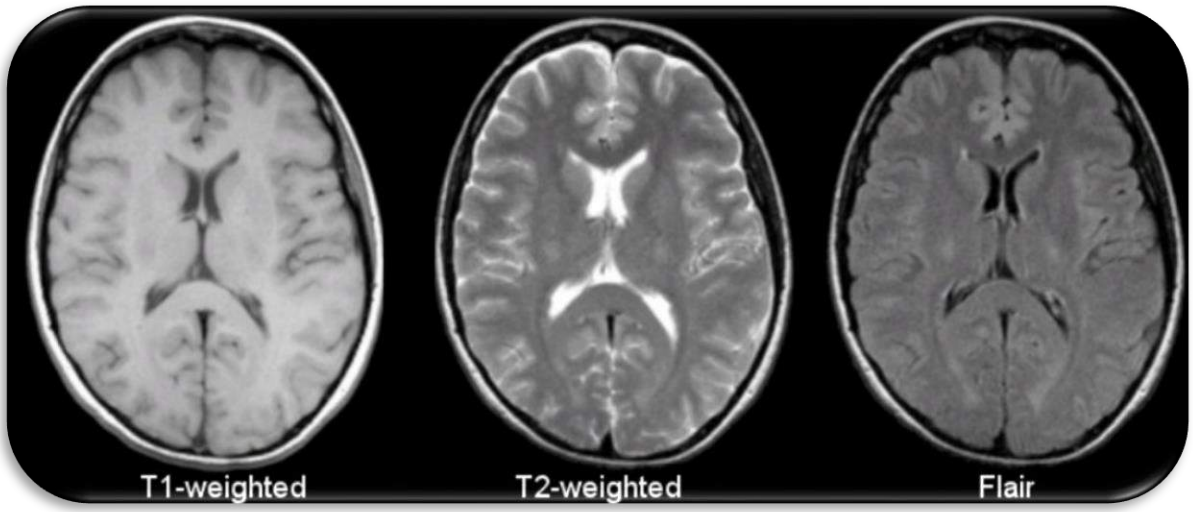


Figure 2: Comparison of T1 vs T2 vs Flair
(Image taken from the dataset
https://ndownloader.figshare.com/articles/1512427/versions/5)

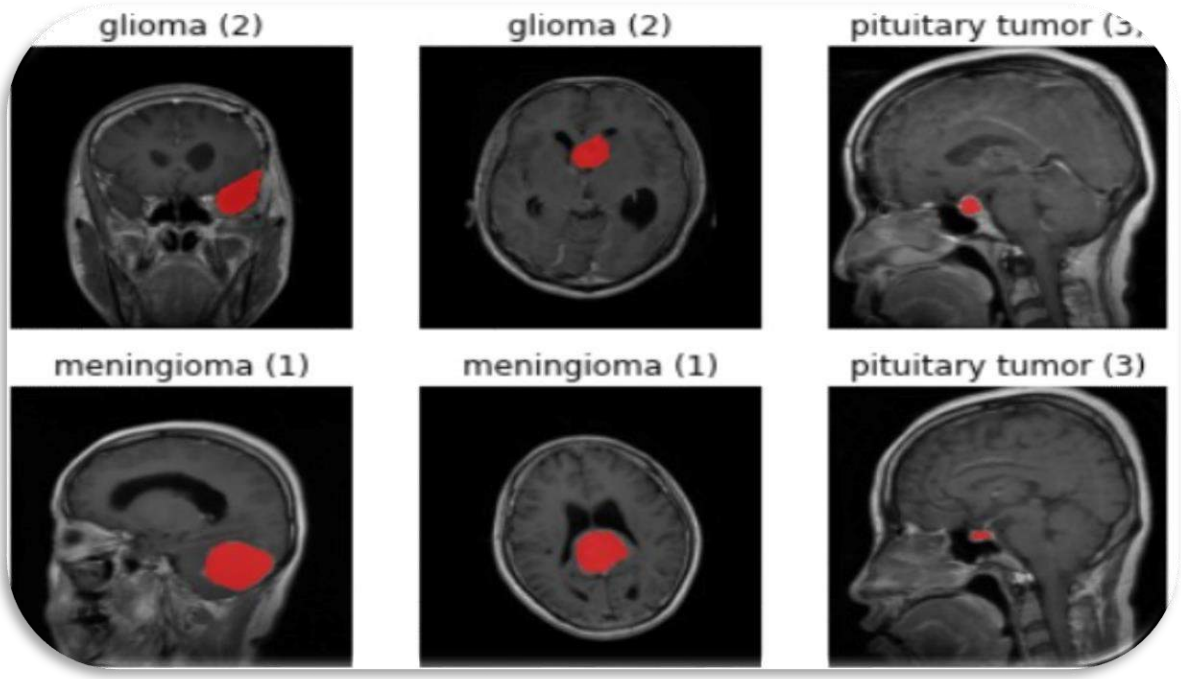


Figure 3: The 3 different kinds of tumor in the given dataset

(Image taken from the dataset
https://ndownloader.figshare.com/articles/1512427/versions/5)

Data Pre-processing

Data pre-processing transforms the data into a format that is more easily and effectively processed in data mining and machine learning.

There are several different tools and methods used for pre-processing data, including the following:

- sampling, which selects a representative subset from a large population of data.
- transformation, which manipulates raw data to produce a single input.
- denoising, which removes noise from data.
- imputation, which synthesizes statistically relevant data for missing values.
- normalization which organizes data for more efficient access.
- feature extraction, which pulls out a relevant feature subset that is significant in a particular context.

Image Segmentation

Image segmentation is a method in which a digital image is broken down into various subgroups called Image segments which helps in reducing the complexity of the image to make further processing or analysis of the image simpler. Segmentation in easy words is assigning labels to pixels. Segmentation has an important role in medical imaging as it helps in extracting the organ of interest.

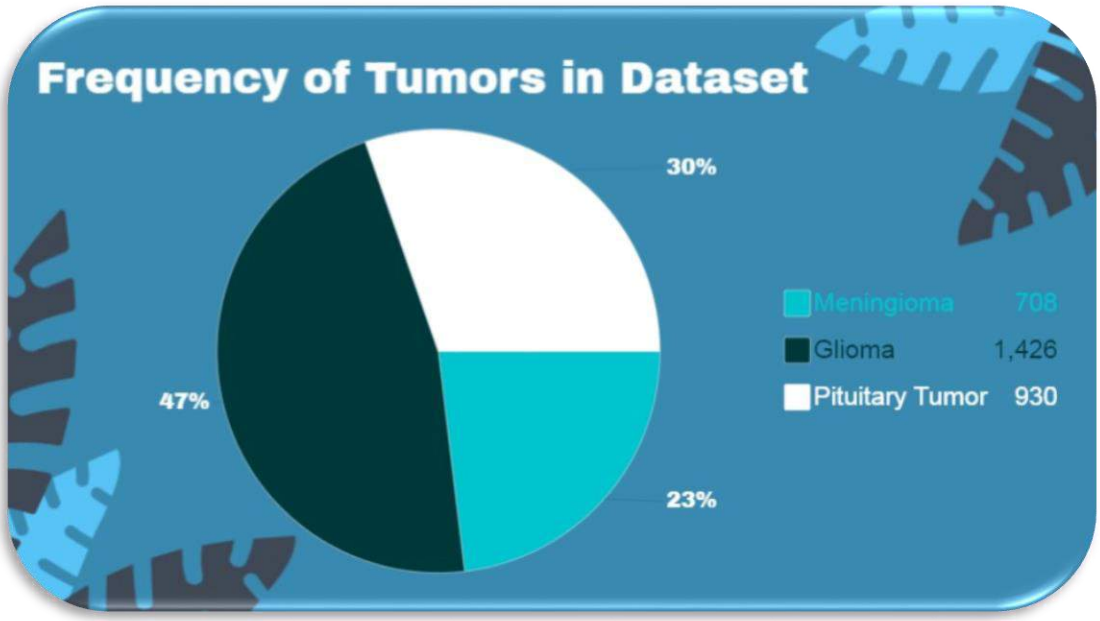


Figure 4: Frequency of tumors in dataset

CNN

Convolutional neural networks (CNNs) are deep neural networks that have the capability to classify and segment images. CNNs can be trained using supervised or unsupervised machine learning methods, depending on what you want them to do. CNN architectures for classification and segmentation include a variety of different layers with specific purposes, such as a convolutional layer, pooling layer, fully connected layers, dropout layers, etc.

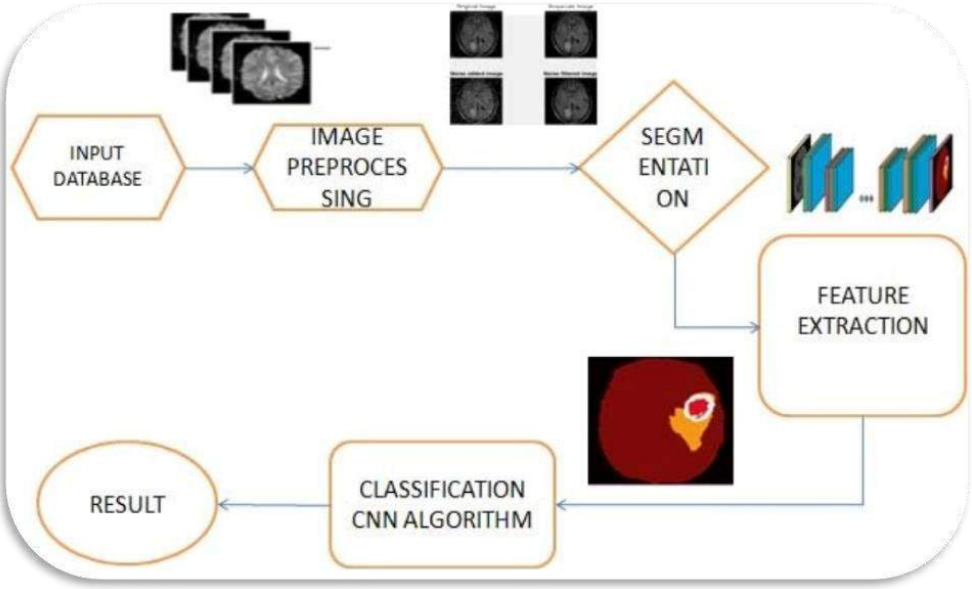


Figure 5: CNN Architecture

(Image Source: https://www.researchgate.net/figure/System-Architecture-Brain-Tumor-Segmentation-using-CNN-is-done-mainly-by-using-three_fig1_344057933)

U-Net Architecture

U-Net is a convolutional neural network architecture that expanded with few changes in the CNN architecture. It was invented to deal with biomedical images where the target is not only to classify whether there is an infection or not but also to identify the area of infection.

- The task in image segmentation is to take an image and divide it into several smaller fragments. These fragments or these multiple segments produced will help with the computation of image segmentation tasks.
- For image segmentation tasks, another essential requirement is the use of masks. With the help of masking, which is basically a binary image consisting of zero or non-zero values, we can obtain the desired result required for the segmentation task.
- With this U-Net architecture, the segmentation of images of sizes 512X512 can be computed with a modern GPU within small amounts of time.

The intent of the U-Net is to capture both the features of the context as well as the localization. This process is completed successfully by the type of architecture built. The main idea of the implementation is to utilize successive contracting layers, which are immediately followed by the up-sampling operators for achieving higher resolution outputs on the input images.

M-Net Architecture

M-Net is inspired by the U-net architecture. Although, both give good performance for their respective tasks, they are not appropriate for segmentation of MRI volume of size 256x256x256, as 2D U-net does not utilize any 3D information, while 3D U-net does it but at the cost of a high memory (10 GB) requirement for a small input of 200x200x50.

The latter is a bottleneck when working with MRI volume of size 256x256x256 as maximum memory available in most advanced GPU is only 12 GB. For addressing this issue of memory constraint, while still utilizing necessary information for MRI segmentation, in a novel way in the already proposed CNN architecture, M-Net is used.

3.2. Development Environment

Software Requirements

- **Python**
  Python is an interpreted, high-level, general purpose programming language created by Guido Van Rossum. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed, and garbage collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

- **NumPy**
  NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.
  It contains various features including these important ones:
  - A powerful N-dimensional array object
  - Sophisticated (broadcasting) functions

- **Keras**
  Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or Plaid ML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

- **PIP**
  It is the package management system used to install and manage software packages written in Python.

- **TensorFlow**
  Tensor flow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

- **Google Colab**
  Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.

- **Matplotlib**
  Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. The . mat extension are files that are in the binary data container format that the MATLAB program uses. The extension was developed by MathWorks, and MAT files are categorized as data files that include variables, functions, arrays and other information.

- **Hdf5 storage**
  HDF5 is a compressed format that is designed to support large, heterogeneous, and complex datasets. Supports Data Slicing: "Data slicing", or extracting portions of the dataset as needed for analysis, means large files don't need to be completely read into the computer's memory or RAM. Version 7.3 of MAT-files uses HDF5 format, this format has a significant storage overhead to describe the contents of the file, especially so for complex nested cellarrays and structures. Its main advantage over previous versions of MAT-files is that it allows storing data larger than 2GB on 64-bit systems.

- **OpenCV**
  OpenCV (Open-source computer vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by willow garage then It sees (which was later acquired by Intel). The library is cross platform and free for use under the open-source BSD license. OpenCV supports some models from deep learning frameworks like TensorFlow, Torch, PyTorch (after converting to an ONNX model) and Caffe according to a defined list of supported layers. It promotes Open Vision Capsules. which is a portable format, compatible with all other formats.

Hardware Configuration

- Processor: Intel core i5 or above.
- 64-bit, quad-core, 2.5 GHz minimum per core
- Ram: 12.68 GB
- Hard disk: 78 GB
- GPU: NVIDIA Tesla K80
- Display: Dual XGA (1024 x 768) or higher resolution monitors
- Operating system: Windows

3.3. Methodology

CNN

CNN stands for Convolutional Neural Network, is used for the classification of images. The input images which are used in CNN are of 3 dimensions i.e., height, width, and number of channels. First two dimensions tell us the image resolution. And third dimension represents the number of channels (RGB) or intensity values for red, green, and blue colors. Usually, images which are fed into the neural network are reduced in dimensions which reduce the processing time and avoid the problem of under fitting.

Convolutional Neural Networks (CNNs) are multi-layer feedforward networks specifically designed to recognize features in 2-dimensional image data. CNNs are primarily used for 2D image recognition, so we will illustrate their architecture on a 2D rectangular image consisting of pixels. Each pixel generally carries colour information. Colon can be represented by multiple channels (e.g- 3 RGB channels). The essence of CNNs is the convolutions. The main trick with convolutional networks that avoids the problem of too many parameters is sparse connections

U Net Architecture

The UNET was developed by Olaf Ronneberger et al. for Bio Medical Image Segmentation. The architecture contains two paths. First path is the contraction path (also called as the encoder) which is used to capture the context in the image. The encoder is just a traditional stack of convolutional and max pooling layers. The second path is the symmetric expanding path (also called as the decoder) which is used to enable precise localization using transposed convolutions. Thus, it is an end-to-end fully convolutional network (FCN), i.e., it only contains Convolutional layers and does not contain any Dense layer because of which it can accept image of any size.

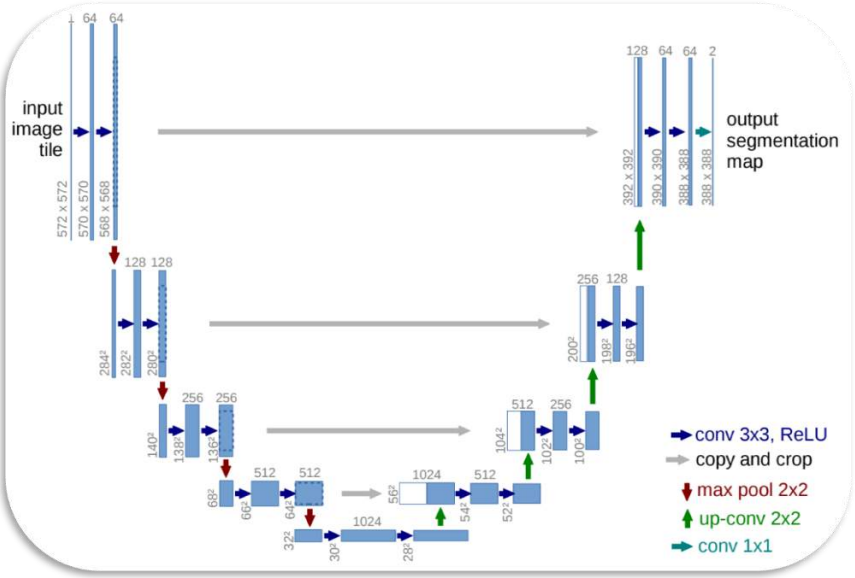


Figure 6: U Net Architecture

(Image Source: https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/)

U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

```
def unet(input_size = (512,512,1)):
    inputs = Input(input_size)
    conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(inputs)
    conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool1)
    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool2)
    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)
    conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool3)
    conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv4)
    drop4 = Dropout(0.5)(conv4)
    pool4 = MaxPooling2D(pool_size=(2, 2))(drop4)

    conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool4)
    conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv5)
    drop5 = Dropout(0.5)(conv5)

    up6 = Conv2D(512, 2, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(drop5))
    merge6 = concatenate([drop4,up6], axis = 3)
    conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge6)
    conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv6)

    up7 = Conv2D(256, 2, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(conv6))
    merge7 = concatenate([conv3,up7], axis = 3)
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge7)
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv7)
```

Figure 7: Code Snippet U-Net Model

M-Net Architecture

M-net has mainly 4 pathways of 2D filters: two main encoding & decoding paths, and two side paths which gives our architecture functionality of deep-supervision. Each pathway has 4 steps. In the encoding path, each step has a cascade of 2D convolution filters of size 3x3 and maxpooling by 2x2, which reduces the size of input by half and allows network to learn contextual information. The decoding layer is identical to encoding layers with one exception: maxpooling is replaced by up sampling layer to double the size of input and recover an output image of original size.

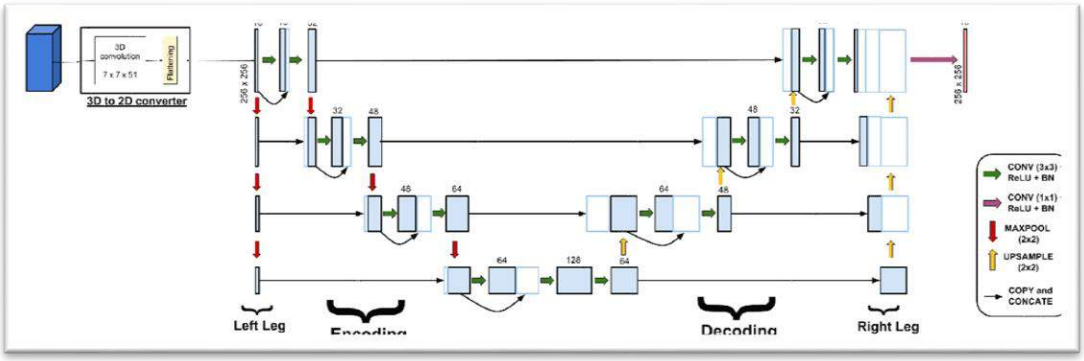


Figure 8: M-Net Architecture

(Image Source: https://www.semanticscholar.org/paper/M-net%3A-A-Convolutional-Neural-Network-for-deep-Mehta-Sivaswamy/fbedd7bb73cca531bb4b4b2f4922aa9db3302214)

```python
def Mnet(size_set=512):
    img_input = Input(shape=(size_set, size_set, 1))

    scale_img_2 = AveragePooling2D(pool_size=(2, 2))(img_input)
    scale_img_3 = AveragePooling2D(pool_size=(2, 2))(scale_img_2)
    scale_img_4 = AveragePooling2D(pool_size=(2, 2))(scale_img_3)

    conv1 = Conv2D(32, (3, 3), padding='same', activation='relu', name='block1_conv1')(img_input)
    conv1 = Conv2D(32, (3, 3), padding='same', activation='relu', name='block1_conv2')(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)

    input2 = Conv2D(64, (3, 3), padding='same', activation='relu', name='block2_input1')(scale_img_2)
    input2 = concatenate([input2, pool1], axis=3)
    conv2 = Conv2D(64, (3, 3), padding='same', activation='relu', name='block2_conv1')(input2)
    conv2 = Conv2D(64, (3, 3), padding='same', activation='relu', name='block2_conv2')(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)

    input3 = Conv2D(128, (3, 3), padding='same', activation='relu', name='block3_input1')(scale_img_3)
    input3 = concatenate([input3, pool2], axis=3)
    conv3 = Conv2D(128, (3, 3), padding='same', activation='relu', name='block3_conv1')(input3)
    conv3 = Conv2D(128, (3, 3), padding='same', activation='relu', name='block3_conv2')(conv3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)

    input4 = Conv2D(256, (3, 3), padding='same', activation='relu', name='block4_input1')(scale_img_4)
    input4 = concatenate([input4, pool3], axis=3)
    conv4 = Conv2D(256, (3, 3), padding='same', activation='relu', name='block4_conv1')(input4)
    conv4 = Conv2D(256, (3, 3), padding='same', activation='relu', name='block4_conv2')(conv4)
    pool4 = MaxPooling2D(pool_size=(2, 2))(conv4)
```

Figure 9: Code Snippet M-Net model

Testing & Training Details of the dataset

The count of different attributes of the dataset, which consists of Training, Testing, Mask and validation Images.

```
number of training examples = 2451
number of development examples = 307
number of test examples = 306
Images_train shape: (2451, 512, 512, 1)
Masks_train shape: (2451, 512, 512, 1)
Images_val (dev) shape: (307, 512, 512, 1)
Masks_val (dev) shape: (307, 512, 512, 1)
Images_test shape: (306, 512, 512, 1)
Masks_test shape: (306, 512, 512, 1)
```

Figure 10: Training and Testing on Dataset

4. Implementation and Result

4.1 Implementation Details

Respective Convolutional layers and Max pooling layers have been implemented.
Each Convolutional layer consists of 3x3 filters and Max pooling layer makes use of 2x2 kernels.
Activation function used = relu
Padding is kept as 'same' which means that it is applied to the input image so that the input image gets fully covered by the filter and specified stride. It is called SAME because, for stride 1 , the output will be the same as the input.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 512, 512, 1)]	0	[]
conv2d (Conv2D)	(None, 512, 512, 64)	640	['input_1[0][0]']
conv2d_1 (Conv2D)	(None, 512, 512, 64)	36928	['conv2d[0][0]']
max_pooling2d (MaxPooling2D)	(None, 256, 256, 64)	0	['conv2d_1[0][0]']
conv2d_2 (Conv2D)	(None, 256, 256, 128)	73856	['max_pooling2d[0][0]']
conv2d_3 (Conv2D)	(None, 256, 256, 128)	147584	['conv2d_2[0][0]']
max_pooling2d_1 (MaxPooling2D)	(None, 128, 128, 128)	0	['conv2d_3[0][0]']
conv2d_4 (Conv2D)	(None, 128, 128, 256)	295168	['max_pooling2d_1[0][0]']
conv2d_5 (Conv2D)	(None, 128, 128, 256)	590080	['conv2d_4[0][0]']
max_pooling2d_2 (MaxPooling2D)	(None, 64, 64, 256)	0	['conv2d_5[0][0]']

Figure 11: U Net Model Components

Respective Convolutional layers and Max pooling layers have been implemented.
Each pathway has 4 steps. In the encoding path, each step has a cascade of 2D convolution filters of size 3x3 and maxpooling by 2x2, which reduces the size of input by half and allows network to learn contextual information

Layer (type)	Output Shape	Param #	Connected to
input_2 (InputLayer)	[(None, 512, 512, 1)]	0	[]
block1_conv1 (Conv2D)	(None, 512, 512, 32)	320	['input_2[0][0]']
average_pooling2d_3 (AveragePo oling2D)	(None, 256, 256, 1)	0	['input_2[0][0]']
block1_conv2 (Conv2D)	(None, 512, 512, 32)	9248	['block1_conv1[0][0]']
block2_input1 (Conv2D)	(None, 256, 256, 64)	640	['average_pooling2d_3[0][0]']
max_pooling2d_4 (MaxPooling2D)	(None, 256, 256, 32)	0	['block1_conv2[0][0]']
concatenate_7 (Concatenate)	(None, 256, 256, 96)	0	['block2_input1[0][0]', 'max_pooling2d_4[0][0]']
block2_conv1 (Conv2D)	(None, 256, 256, 64)	55360	['concatenate_7[0][0]']
average_pooling2d_4 (AveragePo oling2D)	(None, 128, 128, 1)	0	['average_pooling2d_3[0][0]']
block2_conv2 (Conv2D)	(None, 256, 256, 64)	36928	['block2_conv1[0][0]']

Figure 12: M Net Model Components

Performance Metrics

- **Binary Cross Entropy Loss Function**

Binary cross entropy compares each of the predicted probabilities to actual class output which can be either 0 or 1. It then calculates the score that penalizes the probabilities based on the distance from the expected value. That means how close or far from the actual value.

$$\boldsymbol{Loss = -\frac{1}{Output\ size}\sum_{i=1}^{output\ size} y_i * \log \hat{y}_i + (1 - y_i) * \log(1 - \hat{y}_i)}$$

Here y is the referred to as the target value (in our case 0 or 1) and hence it is the predicted value of the pixel.

Output size is the image size (512 x 512)

- **Dice Coefficient**

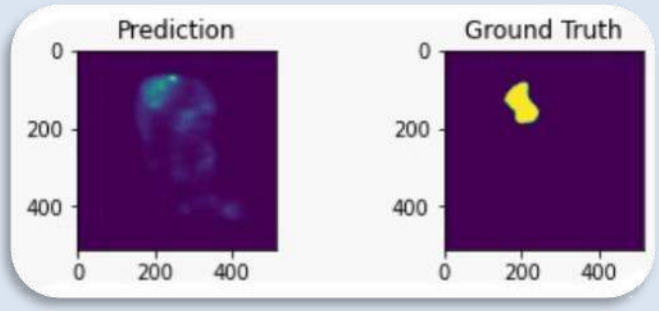
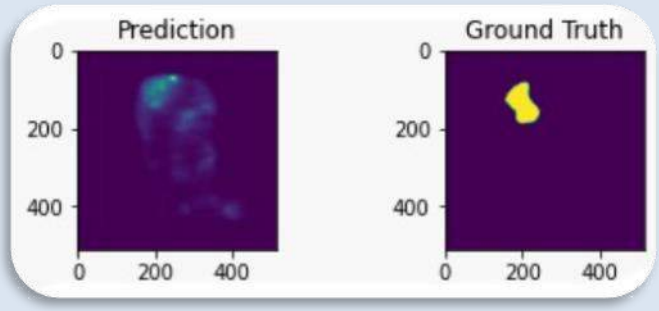
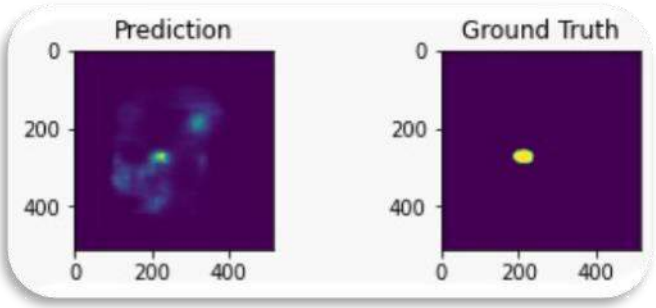
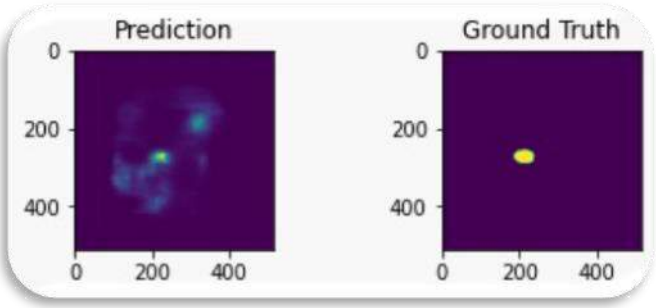
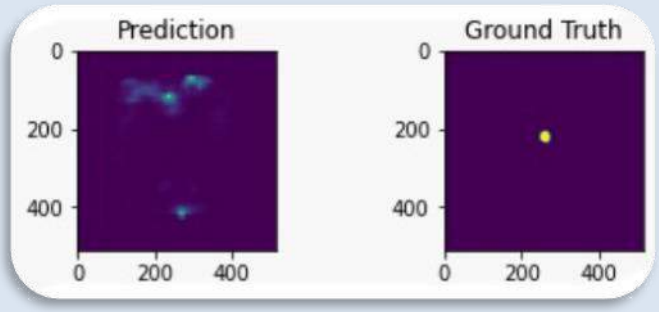
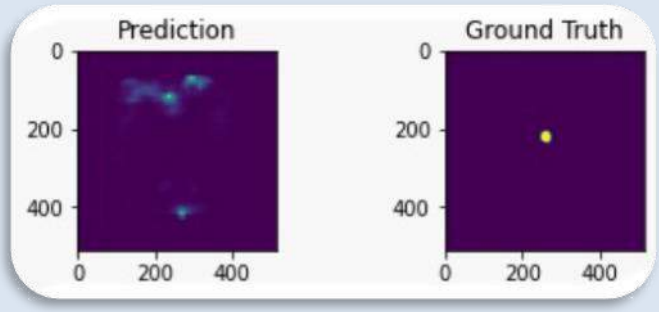
Dice Coefficient is also known as the 'Overlap index'. It is a metric used for validation in domains of medical image segmentation. The pair-wise overlap of the segmentation is calculated by:

$$\boldsymbol{Dice\ Coefficient = \frac{2 * TP}{2 * TP + FP + FN}}$$

- **Dice Loss**

Dice Loss = 1 – Dice Coefficient

4.2 Results and Discussion

U-Net Result Images	M-Net Result Images
	
	
	

Models	Loss	dice_coeff	val_loss	val_dice_coefficient
M-Net	**0.0608**	**0.1046**	**0.0638**	**0.1294**
U-Net	**0.0466**	**0.2993**	**0.0484**	**0.3176**

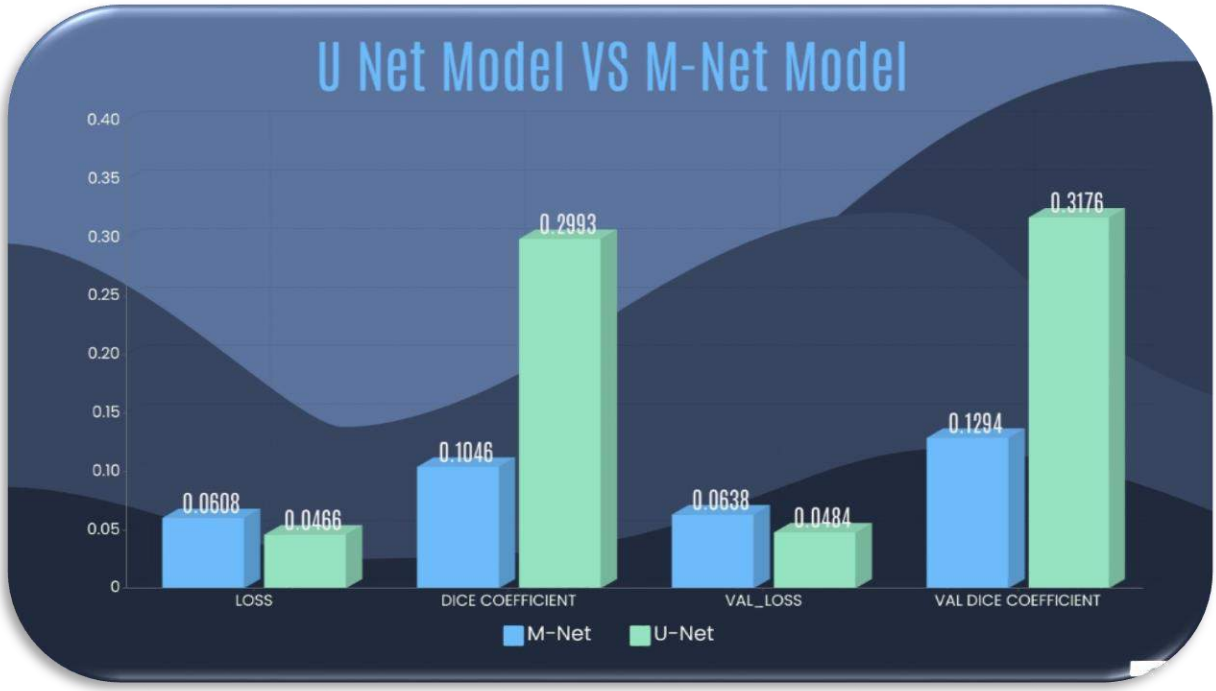


Figure 13: Performance comparison of the models

- The dataset was split into training, testing and validation sets in the ratio-> 80:10:10.
- This resulted in 2451 training images, 306 test images and 306 validation images.
- The model was trained for 5 epochs.
- Google Colab was used to implement the model. It makes use of Intel(R) Xeon(R) CPU @ 2.20GHz, 12 GB RAM and NVIDIA Tesla K80 GPU
- Acquired the right Dataset of MRI images and visualized it correctly with the help of Hdf5 storage for (.mat) formats. Plotted the corresponding masks of the input images and looked at how to go on about constructing the final output images.
- Created a graphical representation of the various types of Tumors present in the dataset and changed the images to gray scale.
- A functional U-Net & M-Net model was successfully built to accurately segment the MRI images.
- Improve the functionality of the model on large datasets and improve the overall accuracy.

4.3 Progress Chart

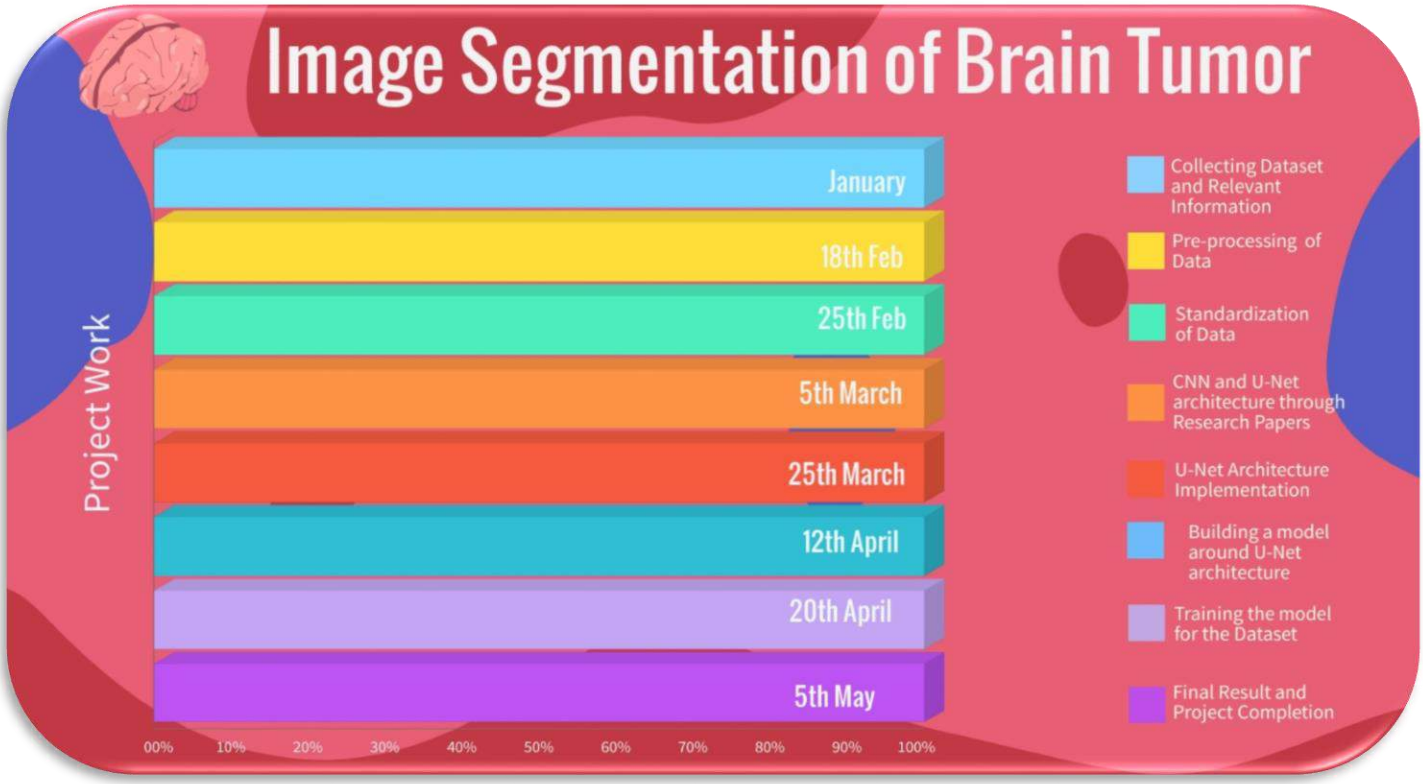


Figure 14: Bar Graph for Project Progress Chart

5. Conclusion and Future Plan

After training both the models on the MRI dataset we conclude that the U-Net model performed better than the M-Net model based on the Dice Coefficient Value of 0.2993. Hence for Biomedical Image Segmentation U-Net model performs with higher accuracy.

The advantage of the M-net is that barring one 3D convolution filter, all other filters are 2D filters which allows end to-end training of the network with considerably low memory requirement (∼5GB).

Future Plans:

- Incorporate 3D volumetric features on MRI images
- Improve the functionality of the model on large datasets and improve the overall accuracy.
- Train the model on higher no.of epochs in order to obtain concrete distinctive features between U-Net and M-Net models used.

References

[1] Priyanka Kadam , S.N. Pawar, 2016, “Brain Tumor Segmentation and It’s Features Extraction by using T2 Weighted Brain MRI”, International Journal of Advanced Research in Computer and Communication Engineering ISO 3297:2007 Certified Vol. 5, Issue 7.

[2] Anil Singh Parihar, 2017,” A Study on Brain Tumor Segmentation Using Convolution Neural Network”, Proceedings of the International Conference on Inventive Computing and Informatics (ICICI 2017) IEEE Xplore Compliant - Part Number: CFP17L34-ART, ISBN: 978-1-5386-4031-9.

[3] Asra Aslam, Ekram Khan, M.M. Sufyan Beg, 2015, “ Improved Edge Detection Algorithm for Brain Tumor Segmentation”, Procedia Computer Science, Volume 58, , Pp 430-437, ISSN 1877-0509

[4] Mahmoud, Dalia & Mohamed, Eltaher. 2012 “Brain Tumor Detection Using Artificial Neural Networks. Journal of Science and Technology” 13. 31-39.

[5] P.S. Mukambika, K Uma Rani, 2017 “Segmentation and Classification of MRI Brain Tumor,” International Research Journal of Engineering and Technology (IRJET), Vol.4, Issue 7, pp. 683 – 688, ISSN: 2395-0056.

[6] Raghav Mehta, “M- Net : A convolutional neural network for deep brain structure segmentation”, Jayanthi Sivaswamy Center for Visual Information Technology (CVIT)

[7] Shruti Jadon, “A survey of loss function for semantic segmentation”.